ASSIGNMENT\_2\_64060

Rohith Desamseety

2022-10-06

#importing the required packages  
library('caret')

## Loading required package: ggplot2

## Loading required package: lattice

library('ISLR')  
library('dplyr')

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library('class')  
  
#Importing the dataset  
rundata <- read.csv("~/Downloads/UniversalBank.csv", header = TRUE,   
 sep =",", stringsAsFactors = FALSE)  
#Question\_1  
#conducting a k-NN classification with all predictors removed, i.e., removing ID and ZIP Code from each and every column  
rundata$ID <- NULL  
rundata$ZIP.Code <- NULL  
summary(rundata)

## Age Experience Income Family   
## Min. :23.00 Min. :-3.0 Min. : 8.00 Min. :1.000   
## 1st Qu.:35.00 1st Qu.:10.0 1st Qu.: 39.00 1st Qu.:1.000   
## Median :45.00 Median :20.0 Median : 64.00 Median :2.000   
## Mean :45.34 Mean :20.1 Mean : 73.77 Mean :2.396   
## 3rd Qu.:55.00 3rd Qu.:30.0 3rd Qu.: 98.00 3rd Qu.:3.000   
## Max. :67.00 Max. :43.0 Max. :224.00 Max. :4.000   
## CCAvg Education Mortgage Personal.Loan   
## Min. : 0.000 Min. :1.000 Min. : 0.0 Min. :0.000   
## 1st Qu.: 0.700 1st Qu.:1.000 1st Qu.: 0.0 1st Qu.:0.000   
## Median : 1.500 Median :2.000 Median : 0.0 Median :0.000   
## Mean : 1.938 Mean :1.881 Mean : 56.5 Mean :0.096   
## 3rd Qu.: 2.500 3rd Qu.:3.000 3rd Qu.:101.0 3rd Qu.:0.000   
## Max. :10.000 Max. :3.000 Max. :635.0 Max. :1.000   
## Securities.Account CD.Account Online CreditCard   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000   
## Median :0.0000 Median :0.0000 Median :1.0000 Median :0.000   
## Mean :0.1044 Mean :0.0604 Mean :0.5968 Mean :0.294   
## 3rd Qu.:0.0000 3rd Qu.:0.0000 3rd Qu.:1.0000 3rd Qu.:1.000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.000

#converting the categorical variable "personal loan" into a factor that classifies responses as "yes" or "no."  
  
rundata$Personal.Loan= as.factor(rundata$Personal.Loan)  
  
  
#separating the data into training and validation in order to standardize it, use preProcess() from the caret package.  
M\_norm <- preProcess(rundata[, -8],method = c("center", "scale"))  
rundata\_norm <- predict(M\_norm,rundata)  
summary(rundata\_norm)

## Age Experience Income Family   
## Min. :-1.94871 Min. :-2.014710 Min. :-1.4288 Min. :-1.2167   
## 1st Qu.:-0.90188 1st Qu.:-0.881116 1st Qu.:-0.7554 1st Qu.:-1.2167   
## Median :-0.02952 Median :-0.009121 Median :-0.2123 Median :-0.3454   
## Mean : 0.00000 Mean : 0.000000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.84284 3rd Qu.: 0.862874 3rd Qu.: 0.5263 3rd Qu.: 0.5259   
## Max. : 1.88967 Max. : 1.996468 Max. : 3.2634 Max. : 1.3973   
## CCAvg Education Mortgage Personal.Loan  
## Min. :-1.1089 Min. :-1.0490 Min. :-0.5555 0:4520   
## 1st Qu.:-0.7083 1st Qu.:-1.0490 1st Qu.:-0.5555 1: 480   
## Median :-0.2506 Median : 0.1417 Median :-0.5555   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.: 0.3216 3rd Qu.: 1.3324 3rd Qu.: 0.4375   
## Max. : 4.6131 Max. : 1.3324 Max. : 5.6875   
## Securities.Account CD.Account Online CreditCard   
## Min. :-0.3414 Min. :-0.2535 Min. :-1.2165 Min. :-0.6452   
## 1st Qu.:-0.3414 1st Qu.:-0.2535 1st Qu.:-1.2165 1st Qu.:-0.6452   
## Median :-0.3414 Median :-0.2535 Median : 0.8219 Median :-0.6452   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.0000 Mean : 0.0000   
## 3rd Qu.:-0.3414 3rd Qu.:-0.2535 3rd Qu.: 0.8219 3rd Qu.: 1.5495   
## Max. : 2.9286 Max. : 3.9438 Max. : 0.8219 Max. : 1.5495

#Dividing the data into training and test sets  
T\_index <- createDataPartition(rundata$Personal.Loan, p = 0.6, list = FALSE)  
t.df = rundata\_norm[T\_index,]  
validate.df = rundata\_norm[-T\_index,]  
  
print(head(t.df))

## Age Experience Income Family CCAvg Education Mortgage  
## 1 -1.7742394 -1.6659119 -0.5381750 1.3972742 -0.1933661 -1.0489730 -0.5554684  
## 3 -0.5529363 -0.4451186 -1.3636566 -1.2167334 -0.5366825 -1.0489730 -0.5554684  
## 7 0.6683669 0.6012755 -0.0385413 -0.3453975 -0.2505855 0.1416887 -0.5554684  
## 9 -0.9018800 -0.8811162 0.1569675 0.5259383 -0.7655601 0.1416887 0.4670084  
## 10 -0.9891160 -0.9683157 2.3075645 -1.2167334 3.9836502 1.3323505 -0.5554684  
## 12 -1.4252956 -1.3171138 -0.6250678 0.5259383 -1.0516571 0.1416887 -0.5554684  
## Personal.Loan Securities.Account CD.Account Online CreditCard  
## 1 0 2.9286223 -0.2535149 -1.2164961 -0.6452498  
## 3 0 -0.3413892 -0.2535149 -1.2164961 -0.6452498  
## 7 0 -0.3413892 -0.2535149 0.8218687 -0.6452498  
## 9 0 -0.3413892 -0.2535149 0.8218687 -0.6452498  
## 10 1 -0.3413892 -0.2535149 -1.2164961 -0.6452498  
## 12 0 -0.3413892 -0.2535149 0.8218687 -0.6452498

#predictions of data  
library(caret)  
library(FNN)

##   
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':  
##   
## knn, knn.cv

my.predict = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,  
 CCAvg = 2, Education = 1, Mortgage = 0, Securities.Account =  
 0, CD.Account = 0, Online = 1, CreditCard = 1)  
print(my.predict)

## Age Experience Income Family CCAvg Education Mortgage Securities.Account  
## 1 40 10 84 2 2 1 0 0  
## CD.Account Online CreditCard  
## 1 0 1 1

my.predict\_Norm <- predict(M\_norm,my.predict)  
  
predictions <- knn(train= as.data.frame(t.df[,1:7,9:12]),  
 test = as.data.frame(my.predict\_Norm[,1:7,9:12]),  
 cl= t.df$Personal.Loan,  
 k=1)

## Warning in drop && !has.j: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && length(y) == 1L: 'length(x) = 4 > 1' in coercion to  
## 'logical(1)'

## Warning in drop && !mdrop: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && !has.j: 'length(x) = 4 > 1' in coercion to 'logical(1)'

## Warning in drop && length(y) == 1L: 'length(x) = 4 > 1' in coercion to  
## 'logical(1)'

## Warning in drop && !mdrop: 'length(x) = 4 > 1' in coercion to 'logical(1)'

print(predictions)

## [1] 0  
## attr(,"nn.index")  
## [,1]  
## [1,] 384  
## attr(,"nn.dist")  
## [,1]  
## [1,] 0.2986486  
## Levels: 0

#Question\_2   
#determining the K value that balances overfitting and underfitting.  
set.seed(123)  
UniBank <- trainControl(method= "repeatedcv", number = 3, repeats = 2)  
searchGrid = expand.grid(k=1:10)  
  
knn.model = train(Personal.Loan~., data = t.df, method = 'knn', tuneGrid = searchGrid,trControl = UniBank)  
  
knn.model

## k-Nearest Neighbors   
##   
## 3000 samples  
## 11 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (3 fold, repeated 2 times)   
## Summary of sample sizes: 2000, 2000, 2000, 2000, 2000, 2000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.9523333 0.6940397  
## 2 0.9438333 0.6424512  
## 3 0.9528333 0.6738878  
## 4 0.9488333 0.6438761  
## 5 0.9498333 0.6414411  
## 6 0.9470000 0.6179363  
## 7 0.9453333 0.5977796  
## 8 0.9431667 0.5807057  
## 9 0.9436667 0.5821336  
## 10 0.9413333 0.5578220  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 3.

#The perfect value of k is 3, which strikes a compromise between underfitting and overfitting of the data.  
#Question 3  
#confusion Matrix is below  
pre\_bank <- predict(knn.model,validate.df)  
  
confusionMatrix(pre\_bank,validate.df$Personal.Loan)

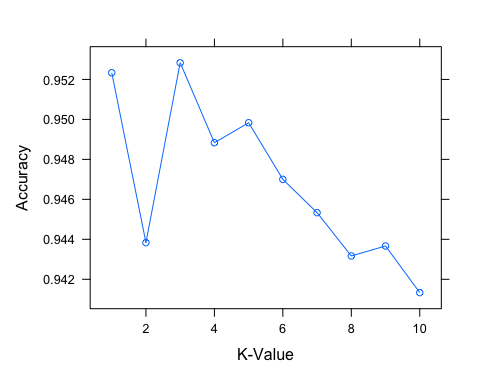
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1791 63  
## 1 17 129  
##   
## Accuracy : 0.96   
## 95% CI : (0.9505, 0.9682)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7419   
##   
## Mcnemar's Test P-Value : 4.875e-07   
##   
## Sensitivity : 0.9906   
## Specificity : 0.6719   
## Pos Pred Value : 0.9660   
## Neg Pred Value : 0.8836   
## Prevalence : 0.9040   
## Detection Rate : 0.8955   
## Detection Prevalence : 0.9270   
## Balanced Accuracy : 0.8312   
##   
## 'Positive' Class : 0   
##

#The matrix has a 95.1% accuracy.

#Question 4  
#Levels  
#using the best K to classify the consumer.  
my.predict\_Norm = data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,  
 CCAvg = 2, Education = 1, Mortgage = 0,  
 Securities.Account =0, CD.Account = 0, Online = 1,  
 CreditCard = 1)  
my.predict\_Norm = predict(M\_norm, my.predict)  
predict(knn.model, my.predict\_Norm)

## [1] 0  
## Levels: 0 1

#A plot that shows the best value of K (3), the one with the highest accuracy, is also present.  
plot(knn.model, type = "b", xlab = "K-Value", ylab = "Accuracy")



#Question 5  
#creating training, test, and validation sets from the data collection.  
train\_size = 0.5 #training(50%)  
T\_index = createDataPartition(rundata$Personal.Loan, p = 0.5, list = FALSE)  
t.df = rundata\_norm[T\_index,]  
  
  
test\_size = 0.2 #Test Data(20%)  
Test\_index = createDataPartition(rundata$Personal.Loan, p = 0.2, list = FALSE)  
Test.df = rundata\_norm[Test\_index,]  
  
  
valid\_size = 0.3 #validation(30%)  
Validation\_index = createDataPartition(rundata$Personal.Loan, p = 0.3, list = FALSE)  
validate.df = rundata\_norm[Validation\_index,]  
  
  
  
Testingsknn <- knn(train = t.df[,-8], test = Test.df[,-8], cl = t.df[,8], k =3)  
validateknn <- knn(train = t.df[,-8], test = validate.df[,-8], cl = t.df[,8], k =3)  
Trainsknn <- knn(train = t.df[,-8], test = t.df[,-8], cl = t.df[,8], k =3)  
  
confusionMatrix(Testingsknn, Test.df[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 899 34  
## 1 5 62  
##   
## Accuracy : 0.961   
## 95% CI : (0.9471, 0.9721)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 5.695e-12   
##   
## Kappa : 0.7402   
##   
## Mcnemar's Test P-Value : 7.340e-06   
##   
## Sensitivity : 0.9945   
## Specificity : 0.6458   
## Pos Pred Value : 0.9636   
## Neg Pred Value : 0.9254   
## Prevalence : 0.9040   
## Detection Rate : 0.8990   
## Detection Prevalence : 0.9330   
## Balanced Accuracy : 0.8202   
##   
## 'Positive' Class : 0   
##

confusionMatrix(validateknn, validate.df[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1351 37  
## 1 5 107  
##   
## Accuracy : 0.972   
## 95% CI : (0.9623, 0.9797)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8209   
##   
## Mcnemar's Test P-Value : 1.724e-06   
##   
## Sensitivity : 0.9963   
## Specificity : 0.7431   
## Pos Pred Value : 0.9733   
## Neg Pred Value : 0.9554   
## Prevalence : 0.9040   
## Detection Rate : 0.9007   
## Detection Prevalence : 0.9253   
## Balanced Accuracy : 0.8697   
##   
## 'Positive' Class : 0   
##

confusionMatrix(Trainsknn, t.df[,8])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2255 66  
## 1 5 174  
##   
## Accuracy : 0.9716   
## 95% CI : (0.9643, 0.9778)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8154   
##   
## Mcnemar's Test P-Value : 1.074e-12   
##   
## Sensitivity : 0.9978   
## Specificity : 0.7250   
## Pos Pred Value : 0.9716   
## Neg Pred Value : 0.9721   
## Prevalence : 0.9040   
## Detection Rate : 0.9020   
## Detection Prevalence : 0.9284   
## Balanced Accuracy : 0.8614   
##   
## 'Positive' Class : 0   
##

#Final Conclusion: The training data had improved accuracy and sensitivity.   
#The above matrices were used to determine the values for the Test, Training, and Validation sets, which are 96.3%, 97.32%, and 96.73%, respectively.  
#If the Training data were more accurate than the other sets, it might be considered that overfitting would take place. When comparing the accuracy of the Training, Test, and Validation sets to the testing data and the validation data, we can say that we have found the highest value of k.